Layering IT Services for a Planetary "Exploration Web"

JPL

Norm Lamarra
Section 366

Oct 29, 2002

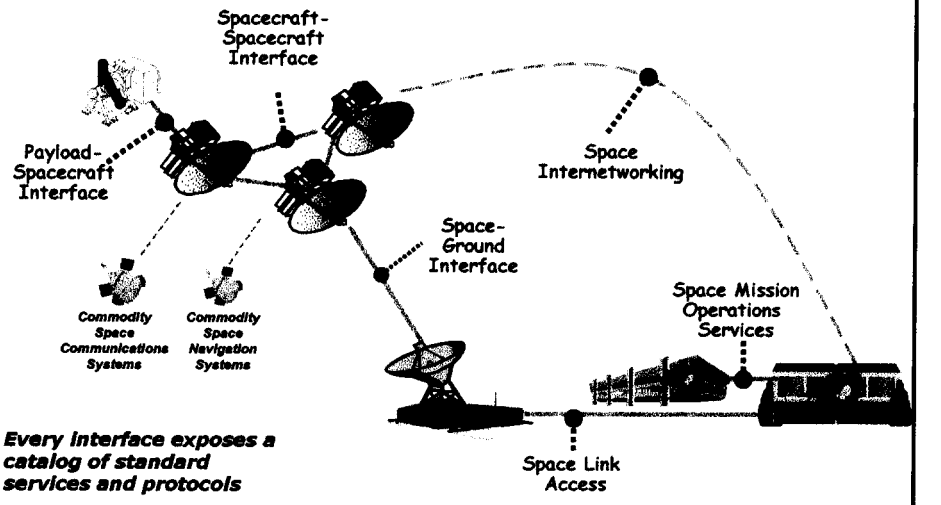
Overview

JPL

- Background & Problem Domain
  - Space Communications Interoperability Points
- Proposed Approach
  - Example Middleware Goals
  - Some Architectural Requirements
  - Solution Approach
- Example Remote Applications
  - FY02 Messaging Prototype
- Benefits of Message-based Middleware
  - Communications, autonomy, automation
- Future Steps
  - Roadmap

Space Communications Interoperability Points

JPL



Source: A. Hooke, NASA/JPL

Example Middleware Goals

JPL

- Improve scientist-instrument connection
  - Large effort and cost of coordinating mission plans
  - Coordination among spacecraft
- Improve average downlinked data value
  - Which bits do we get?
  - Knowledge vs. Information vs. Data
- Enhance remote applications
  - Simplify application interfaces
  - Simplify access to distributed resources
  - Increase robustness (e.g., s/w modification)
- Improve operations (automation & autonomy)
  - Robust execution (e.g., replication, failure detection/recovery)
  - On-board reasoning (e.g., vehicle health, science goals, etc.)

Some Architectural Requirements

JPL

- Support DTN & disconnected operation
  - DTN: delay-tolerant network (latency, bandwidth, etc.)
  - Robust – range of QoS?
  - Event buffering
- Support critical information and distributed object model
  - e.g., vehicle health, "surprising information", etc.
  - Message Prioritization
  - Message Efficiency (low overhead)
- Support asynchronous programming model
  - Support distributed computation (mobile code?) (implicit & explicit)
- Support for global naming (resources, nodes, etc.)
  - Who, what, where
- Provide access to distributed information & services
  - Data repositories, navigation, weather, time, science analysis, etc.
- Lightweight modular reusable infrastructure
  - Allow micro-platforms (sensor webs, etc.) to play (simple client)
  - Able to integrate with flight software (e.g., MDS)

Possible Solution Approach

JPL

- Conceptualize a set of standardized "***shared services***"
  - 3 broad categories: Communications, Storage, Processing
  - Distributed client-server model useful for all 3
    - Make object model highly flexible
    - Make clients as lightweight as possible
    - Simplify server replication (when necessary)
  - Build upon "enhanced" internet-style communication
    - Asynchronous messaging has many advantages
    - Publish/subscribe has further advantages
    - Message prioritization and efficiency are crucial
- Deploy "***layered infrastructure***" incrementally
  - Basic services: Messaging, time, events, security
  - Information services: Remote data management, alarms
  - Higher-level services: navigation, weather, etc.
  - Agent interaction infrastructure (far future)
    - e.g., "autonomous" communication vice "scheduled"

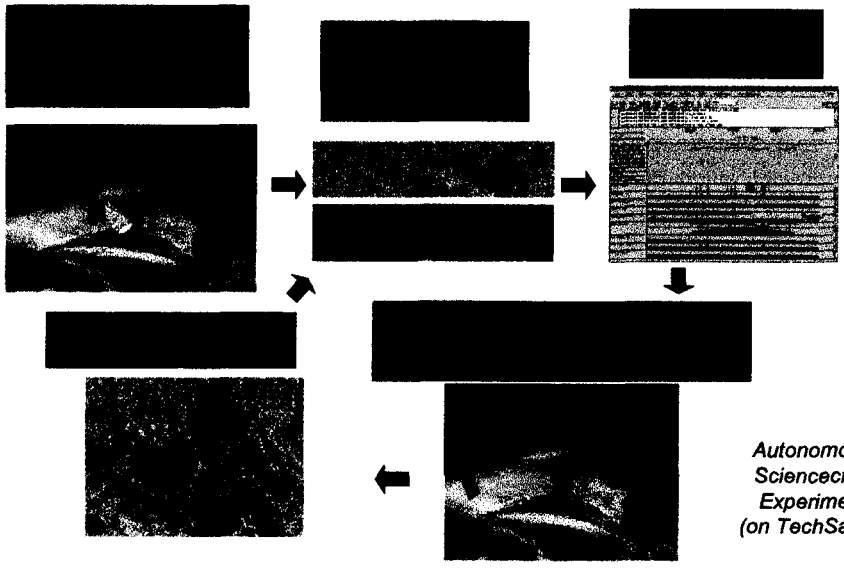
Solution Approach (cont'd.)

JPL

- Issues/Req'ts for shared services
  - **Communications**
    - Tolerate delay, disconnection, b/w limitation
      - Buffered, asynchronous, ...
    - Allow choice of transport protocol
      - Support standards (e.g., CCSDS)
    - Provide QoS (guarantees, reserved b/w, etc.)
    - Allow (dynamic) priorities (inc. time-to-live)
    - Tolerate variety of network topologies (near/far)
  - Storage
    - Provide flexible storage type (e.g., image, meas't, stream)
    - Provide query capability
    - Support management functions (e.g., location, access)
    - Allow transport (e.g., move, replicate)
  - Processing
    - Allow remote processing (like "solver service")
    - Support fault tolerance (e.g., checkpointing, validation)

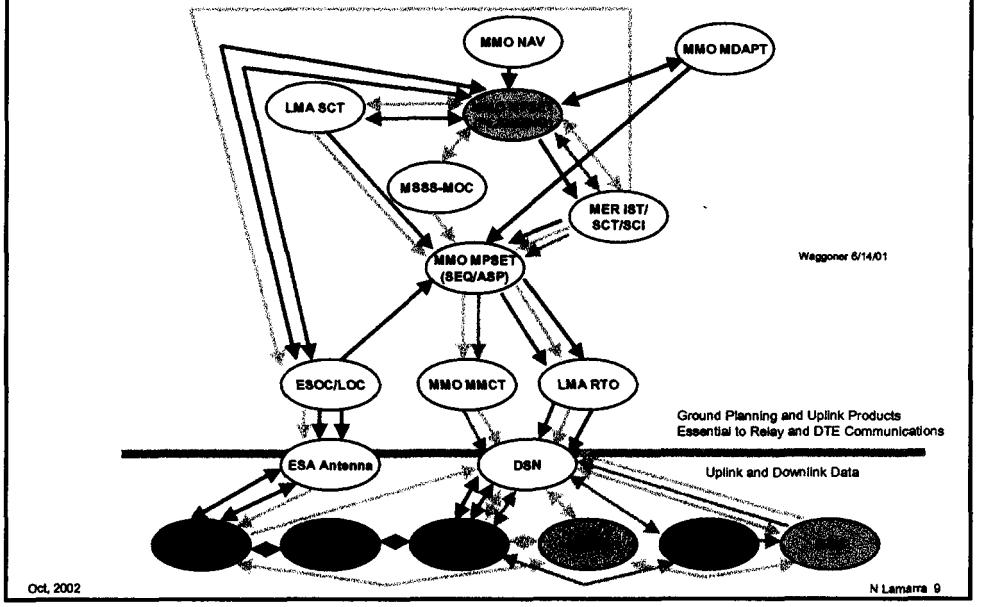
Example: ASE Mission Scenario

JPL



Autonomous Sciencecraft Experiment (on TechSat21)

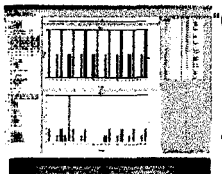
Example Mission-Interaction Dataflow

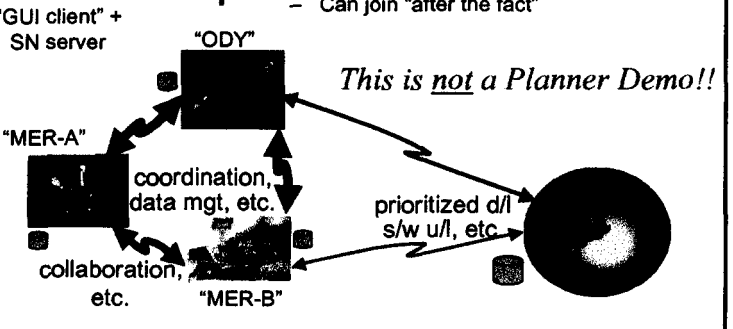


FY02 Messaging Prototype (SharedNet)

Example Scenario	Shows

• Remote Planning Coordination:
  • e.g., MER-A/MER-B/ODY
• Plan change by one affects others
  • Time criticality (view periods)
• Negotiations reach a solution
  • Minimal use of link to Earth
	• "Ad-hoc" remote comms
	  – Robust MOM: buffered, async, QoS,…
	  – Extensible message object model
	• "Subscription" by message type
	  – Simple client (Java API, C++ wrapper)
	• "GUI client" displays filtered traffic
	  – Can join "after the fact"





This is <u>not</u> a Planner Demo!!

Benefits of Message-based Middleware

JPL

- Better use of communication bandwidth
  - "Remote" network is "local" to participants (maybe much more capacity)
    - Redundant, fault-tolerant, dynamic, etc.
  - Much sensor information cannot reach Earth (e.g., MGS d/l <1% data)
    - Could be processed "locally" and shared efficiently
    - Can pick up "waiting" info at later time (e.g., when in view)
  - Sensor networks more easily integrated (esp. different sensors)
    - Flexible message routing and filtering
    - Remove dependence on time, address, platform
- Improve automation
  - "Alerts" and "Events" can trigger procedures
    - e.g., health/status of spacecraft and sensors; automated info
  - Simplify use of on-board & distributed processing
    - Software upload/installation (e.g., fixes to Galileo)
  - Assist failure discovery/recovery
    - Process restart or migration; application reconfiguration
- Assist future autonomy
  - Simpler infrastructure for collaboration (joint planning, etc.)
    - Distributed intelligence
  - More sources of information accessible for decisions
    - e.g., terrain, weather, other off-board sensors

Future Steps

JPL

- Messaging is required for almost all distributed apps
  - MOM provides simple model for application messaging:
    - Other information services can be layered above MOM
      - e.g., distributed data mgt.: access/relay/archive/query
    - Other protocols can be layered below MOM
      - e.g., IP, CCSDS: PROX-1, CFDP
- Improve MOM functionality for space
  - Simplify extending Message Object Model
  - Message forwarding; adaptive operation
  - Verify robustness (disconnection, b/w, etc.)
  - Address CCSDS standards (e.g., SOIF)
  - Enable dynamic installation/removal – "standard services"
- Integrate with other applications
  - Simplify on-board processing (e.g., science extraction)
    - Provide support for dynamic algorithms (e.g., module upload)
  - Address software architecture issues (e.g., MDS, CLARAty)
- Migrate to RTOS on flight hardware
  - e.g., VxWorks on PPC

Roadmap for Space Middleware

JPL

- On-board processing
  - Image/sensor analysis; pattern recognition
- Data services (to in-situ assets)
  - Weather/climate info (e.g., from sensor webs)
  - Position/capabilities/availability of "nearby" assets
- Autonomy/Collaboration
  - In-situ scheduling; distributed operations
    - Sequence generation; science team collaboration
    - Continuous planning; negotiated plans
  - Multi-agent coordination
- Virtual Exploration
  - Telepresence; video; audio
  - Virtual Environments

Complexity